

**METHOD AND APPARATUS FOR AN IMPROVED
BULK READ SOCKET CALL**

BACKGROUND OF THE INVENTION

5

1. Technical Field:

The present invention is directed to a method and apparatus for an improved bulk read socket call.

10 **2. Description of Related Art:**

The Internet has become a significant communication media in the modern world and is enabling the world to migrate to one global data communications system. The Internet uses the Transmission Control Protocol/Internet Protocol (TCP/IP) suite of protocols to provide a common communications mechanism for computers, and other data transmission devices, to communicate with one another.

Communication with applications running on a server is typically performed using ports and addresses assigned to the application and the server apparatus. A port may be a physical port or a logical port. A physical port is a pathway into and out of a computer or a network device such as a switch or router. For example, the serial and parallel ports on a personal computer are external sockets for plugging in communications lines, modems and printers. Every network adapter has a port (Ethernet, Token Ring, etc.) for connection to the local area network (LAN). Any device that transmits and receives data implies an available port to connect to each line.

Docket No. AUS920010389US1

A logical port is a number assigned to an application running on the server by which the application can be identified. While a server may have a single physical port, the server may make use of a plurality of logical ports. The combination of a logical port identifier and the address of the server apparatus is referred to as a socket.

The address of the server is a network address that identifies the server in the network and how to route data to a particular physical port of the server through the network. The address may take the form of a Uniform Resource Locator (URL), or in the case of the Internet, an Internet Protocol (IP) address such as 205.15.01.01, or the like. The address is included in headers of data packets transmitted by a device. The data packets are routed through the network from device to device by reading the header of the data packet and determining how to route the data packet to its intended destination based on the address.

Once the data packet arrives at the intended destination server apparatus, the server determines the destination application based on the logical port identifier included in the header information of the data packet. A data packet may be directed to a particular logical port by including the logical port identifier in its header information.

An application on a server "listens" to a logical port by retrieving data having a logical port identifier

Docket No. AUS920010389US1

that identifies the logical port associated with that application. The application will take the data directed to its logical port and place it in a queue for the application. In this way, data may be routed through a
5 network to a server apparatus and provided to a particular application on the server apparatus for processing.

The TCP/IP protocol provides various socket functions that may be used in the handling of data as the
10 data is transported to and from an application through the socket. One such function that is typically used by FTP file and print services is the `recv(int sock, (void *) buffer, int flags)` read function. This read function further has a known feature `MSG_WAITALL` that allows an
15 application to read a large amount of data at one time as a bulk read instead of reading a large amount of data doing multiple calls of the `recv()` function.

For example, assume that a user wishes to read data in bulk units of 60,000 bytes. Using the `MSG_WAITALL`
20 bulk read feature of the `recv()` function, each time data is stored in the receive socket buffer, the `recv()` bulk read function is awakened. The `recv()` examines the receive socket buffer to determine if 60,000 bytes are in the receive socket buffer. If not, the `recv()` goes back
25 to sleep and waits until another amount of data is stored in the receive socket buffer when it will again be awakened. If there is 60,000 bytes in the receive socket buffer, this amount of data is read from the receive socket buffer and provided to the calling application.

Docket No. AUS920010389US1

With this bulk read function, a single call to the
recv() function is made rather than multiple calls and
thus, the overhead associated with the extra system call
execution is avoided. However, this feature, when used
5 with TCP sessions has many limitations.

First, even though the recv() function waits for the
full amount of data the user has requested, the TCP wakes
up the blocked thread, i.e. the thread from the calling
application that calls recv(), each time a data segment
10 arrives. The thread wakes up and checks if the full data
has arrived. If not, it goes back to sleep again. For a
64 Kb recv() function call, for example, receiving 1460
byte Maximum Segment Size (MSS) TCP segments, this would
result in approximately 43 unnecessary wakeups of the
15 thread and the associated overhead.

Second, when MSG_WAITALL is used, TCP
acknowledgments are delayed up to 200 milliseconds. The
reason for this is that acknowledgments are triggered by
the application reading at least 2 MSS worth of data from
the receive socket buffer. However, when the MSG_WAITALL
20 feature is used, since the data remains in the receive
socket buffer until the user requested amount of data is
gathered in the buffer, acknowledgments will not be sent
until the delayed acknowledgment timer expires. The
25 delayed acknowledgment timer is a timer that delays the
sending of an acknowledgment of receipt of data up to 200
milliseconds in anticipation of sending the
acknowledgment with data that needs to be sent in the
reverse direction. Delaying the acknowledgments so much
30 causes a number of problems.

Docket No. AUS920010389US1

For example, the TCP's congestion control and avoidance schemes, such as slow start and the like, depend heavily on incoming acknowledgments. Slow start, for example, is the phase of data transmission in which
5 the sending computing device sends data slowly and increases the data flow on the arrival of each acknowledgment from the receiving computing device.

Thus, for example, if the acknowledgments are delayed, during the slow start phase, the congestion
10 window will open up very slowly, i.e. the data flow will increase very slowly. The congestion window is a measurement of the amount of data that a sender may send to a receiving computing device and avoid causing congestion in the data network.

15 In addition, the fast recovery mechanism after detecting packet loss recovers solely depending on the arrival of acknowledgments. For example, assume that a receiver is waiting for 32 Kb of data on a recv() function call with MSG_WAITALL set and a sender's
20 congestion window is currently 22 segments, i.e. TCP packets, of 1460 bytes. If a data packet gets dropped, the fast retransmit algorithm retransmits the dropped segment after receiving three duplicate acknowledgments but also halves the congestion window to 11 segments to
25 thereby slow down the data traffic in the network.

Assuming, for example, a single packet drop, the receiver would acknowledge all 22 segments on receiving the dropped segment and the recv() function should complete. However, for the next "send", the sender is
30 allowed to send only 11 segments whereas the receiver is

Docket No. AUS920010389US1

waiting for the full 32 Kb. There will now be a pause until the 200 millisecond delayed acknowledgment timer expires. Then the TCP would acknowledge the 11 segments. Now the sender can send the next 11 segments. On

5 receiving the next 11 segments, the `recv()` function would also complete. However, since the sender is now in a fast recover phase, the congestion window opens up by only 1/11th of the segment size per acknowledgment. Therefore, for the next `recv()` call, the same 200
10 millisecond delay occurs. This continues until the congestion window grows back to 22 segments. The above example considers only one segment loss for this duration. The situation is considerably worse when multiple packet drops occur.

15 Third, when the `MSG_WAITALL` flag is used, since the receiver's advertised window, i.e. the size of the receivers TCP buffer, keeps reducing until the `recv()` function gets the full data requested by the user, a situation may occur where the sender hits the persist
20 timer delays (minimum of 5 seconds in most implementations). The persist timer delays are the delays perceived by the sending computing system due to probing of the receiving computing system to determine when the receiving computing system TCP buffer is no
25 longer full. This is caused due to the fact that TCP is byte oriented and not message oriented.

A 32 Kb message written by the sender gets packetized and depacketized by TCP in a manner determined by TCP. When the receiver side window reduces to a value
30 less than the MSS used by the connection, the sender

Docket No. AUS920010389US1

defers sending this small amount of data (if it has enough data queued in its send buffer) until the receiver opens up its window because the sender thinks the receiver is busy and is slow to read the data. This may
5 not be the case, however, because the receiver might actually be waiting on the `recv()` function for just this small piece of data to make the 32 Kb that the user has requested.

Thus, it would be beneficial to have a method and
10 apparatus for an improved bulk read socket call that avoids the drawbacks of the prior art outlined above.

SUMMARY OF THE INVENTION

The present invention provides an apparatus and method for an improved bulk read socket call are provided. With the apparatus and method of the present invention, a new field, `so_rcvlen`, is added to the socket structure that identifies the bulk read size requested by the user. The kernel of the prior art `recv()` function is also modified so that it sets the `so_rcvlen` to the size requested by the user prior to the `recv()` function going to sleep and waiting for the full data size requested by the user. In addition, a new flag, `SP_MSGWAITALL`, is also provided in the socket structure.

In the TCP input processing of the present invention, when data is received for a particular socket, the current setting of the `SP_MSGWAITALL` is checked. If the `SP_MSGWAITALL` flag is set, it is determined whether the amount of data stored in the socket receive buffer is less than the value of `so_rcvlen`. If not, the TCP input processing does not wake up the `recv()` thread. However, for every alternate segment, the TCP input processing sends back an acknowledgment (ACK).

In the TCP output processing of the present invention, when the `SP_MSGWAITALL` flag is set and the amount of data in the socket receive buffer is less than `so_rcvlen`, the full window is advertised. Once the TCP input processing determines that there is at least an amount of data in the socket receive buffer equal to the value of `so_rcvlen`, the TCP input processing will wake up the `recv()` thread and the `SP_MSGWAITALL` flag is reset.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented;

Figure 2 is a block diagram of a data processing system that may be implemented as a server in accordance with a preferred embodiment of the present invention;

Figure 3 is a block diagram illustrating a data processing system in which the present invention may be implemented; and

Figure 4 is a flowchart outlining an exemplary operation of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures, **Figure 1** depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented. Network data processing system **100** is a network of computers in which the present invention may be implemented. Network data processing system **100** contains a network **102**, which is the medium used to provide communications links between various devices and computers connected together within network data processing system **100**. Network **102** may include connections, such as wire, wireless communication links, or fiber optic cables.

In the depicted example, a server **104** is connected to network **102** along with storage unit **106**. In addition, clients **108**, **110**, and **112** also are connected to network **102**. These clients **108**, **110**, and **112** may be, for example, personal computers or network computers. In the depicted example, server **104** provides data, such as boot files, operating system images, and applications to clients **108-112**. Clients **108**, **110**, and **112** are clients to server **104**. Network data processing system **100** may include additional servers, clients, and other devices not shown. In the depicted example, network data processing system **100** is the Internet with network **102** representing a worldwide collection of networks and gateways that use the TCP/IP suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host computers, consisting

of thousands of commercial, government, educational and other computer systems that route data and messages. Of course, network data processing system **100** also may be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). **Figure 1** is intended as an example, and not as an architectural limitation for the present invention.

Referring to **Figure 2**, a block diagram of a data processing system that may be implemented as a server, such as servers **104**, **114**, **118** in **Figure 1B**, is depicted in accordance with a preferred embodiment of the present invention. Data processing system **200** may be a symmetric multiprocessor (SMP) system including a plurality of processors **202** and **204** connected to system bus **206**. Alternatively, a single processor system may be employed. Also connected to system bus **206** is memory controller/cache **208**, which provides an interface to local memory **209**. I/O bus bridge **210** is connected to system bus **206** and provides an interface to I/O bus **212**. Memory controller/cache **208** and I/O bus bridge **210** may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge **214** connected to I/O bus **212** provides an interface to PCI local bus **216**. A number of modems may be connected to PCI bus **216**. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communications links to network computers **108-112** in **Figure 1B** may be provided through modem **218** and network

Docket No. AUS920010389US1

adapter **220** connected to PCI local bus **216** through add-in boards.

Additional PCI bus bridges **222** and **224** provide interfaces for additional PCI buses **226** and **228**, from which additional modems or network adapters may be supported. In this manner, data processing system **200** allows connections to multiple network computers. A memory-mapped graphics adapter **230** and hard disk **232** may also be connected to I/O bus **212** as depicted, either directly or indirectly.

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figure 2** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

The data processing system depicted in **Figure 2** may be, for example, an IBM RISC/System 6000 system, a product of International Business Machines Corporation in Armonk, New York, running the Advanced Interactive Executive (AIX) operating system.

With reference now to **Figure 3**, a block diagram illustrating a data processing system is depicted in which the present invention may be implemented. Data processing system **300** is an example of a client computer. Data processing system **300** employs a peripheral component interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures such as Accelerated Graphics Port (AGP) and Industry Standard Architecture (ISA) may be used.

Docket No. AUS920010389US1

Processor **302** and main memory **304** are connected to PCI local bus **306** through PCI bridge **308**. PCI bridge **308** also may include an integrated memory controller and cache memory for processor **302**. Additional connections to PCI local bus **306** may be made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter **310**, SCSI host bus adapter **312**, and expansion bus interface **314** are connected to PCI local bus **306** by direct component connection. In contrast, audio adapter **316**, graphics adapter **318**, and audio/video adapter **319** are connected to PCI local bus **306** by add-in boards inserted into expansion slots. Expansion bus interface **314** provides a connection for a keyboard and mouse adapter **320**, modem **322**, and additional memory **324**. Small computer system interface (SCSI) host bus adapter **312** provides a connection for hard disk drive **326**, tape drive **328**, and CD-ROM drive **330**. Typical PCI local bus implementations will support three or four PCI expansion slots or add-in connectors.

An operating system runs on processor **302** and is used to coordinate and provide control of various components within data processing system **300** in **Figure 3**. The operating system may be a commercially available operating system, such as Windows 2000, which is available from Microsoft Corporation. An object oriented programming system such as Java may run in conjunction with the operating system and provide calls to the operating system from Java programs or applications executing on data processing system **300**. "Java" is a

trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented operating system, and applications or programs are located on storage devices, such as hard disk drive **326**, and may be loaded
5 into main memory **304** for execution by processor **302**.

Those of ordinary skill in the art will appreciate that the hardware in **Figure 3** may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash ROM (or equivalent nonvolatile
10 memory) or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in **Figure 3**. Also, the processes of the present invention may be applied to a multiprocessor data processing system.

15 As another example, data processing system **300** may be a stand-alone system configured to be bootable without relying on some type of network communication interface, whether or not data processing system **300** comprises some type of network communication interface. As a further
20 example, data processing system **300** may be a Personal Digital Assistant (PDA) device, which is configured with ROM and/or flash ROM in order to provide non-volatile memory for storing operating system files and/or user-generated data.

25 The depicted example in **Figure 3** and above-described examples are not meant to imply architectural limitations. For example, data processing system **300** also may be a notebook computer or hand held computer in addition to taking the form of a PDA. Data processing
30 system **300** also may be a kiosk or a Web appliance.

Docket No. AUS920010389US1

The present invention provides a mechanism in the application layer and TCP layer of the standard TCP/IP stack that moves the monitoring function of the socket receive buffer from the `recv()` function to the TCP processing. In moving the monitoring function to the TCP processing, much of the overhead associated with waking up the `recv()` thread and putting it back to sleep in the prior art is eliminated. In addition, the problems with regard to not sending out acknowledgment messages are also overcome by the present invention.

With the present invention, a new field, `so_rcvlen` is added to the socket structure, i.e. the kernel structure per socket that contains the socket send and receive buffer queues, points to the communication protocol (TCP, UDP, UNIX, etc.) private data, and contains state and error information. This new field identifies the bulk read size requested by the user. The kernel of the `recv()` function, i.e. the fundamental part of a the `recv()` function that resides in memory at all times, is also modified so that it sets the `so_rcvlen` to the size requested by the user prior to the `recv()` function going to sleep and waiting for the full data size requested by the user.

In addition, a new flag, `SP_MSGWAITALL`, is provided in the field of the socket structure that stores flags. The `SP_MSGWAITALL` flag is used to identify when the functions of the present invention are to be implemented, as described hereafter. Thus, the application layer of the present invention is modified from the application layer of the prior art by providing a new field in the

Docket No. AUS920010389US1

socket structure for identifying the requested bulk read size and by providing a new flag for identifying when the functions of the present invention are to be implemented.

With regard to the TCP layer, the present invention
5 modifies the TCP layer of the prior art by providing additional functionality for performing functions based on the settings of the `so_rcvlen`, `SP_MSGWAITALL`, and the amount of data stored in the socket receive buffer, as detailed hereafter.

10 At different TCP states, TCP is required (under RFC 793), to handle incoming segments differently. TCP state transitions occur based on events such as incoming TCP segments. TCP input processing is the TCP code that is implemented to handle a TCP segment that arrives at a TCP
15 enabled device. Some of the different tasks performed by TCP input processing include validating an incoming segment for data corruption, determining to which socket, i.e. which user application, this segment belongs, delivering data to the user application, performing state
20 transitions (e.g., when the incoming segment indicates the connection needs to be closed), and the like.

With the present invention, in the TCP input processing, the following new functions are provided. When data is received for a particular socket, the
25 current setting of the `SP_MSGWAITALL` is checked. If the `SP_MSGWAITALL` flag is set on the socket, a determination is made as to whether the amount of data stored in the socket receive buffer is less than the value of `so_rcvlen`. If the amount of data stored in the socket
30 receive buffer is less than the value of `so_rcvlen`, the

Docket No. AUS920010389US1

TCP input processing does not wake up the `recv()` thread. However, for every alternate segment, the TCP input processing sends back an acknowledgment (ACK).

Thus, the overhead of waking up the `recv()` thread and putting it back to sleep again, because there is not enough data in the receive socket buffer, is avoided. Additionally, the present invention sends out acknowledgments for every alternate segment and thus, the problems outlined above with regard to the prior art not sending out acknowledgments are avoided.

In the TCP output processing, the following new functions are provided. When the `SP_MSGWAITALL` flag is set in the socket and the amount of data in the socket receive buffer is less than `so_rcvlen`, the full window is advertised. The full window is the window size advertised by the receiving computing device to the sending computing device at connection setup time. By advertising the full window, the receiving computing device informs the sending computing device of the available socket buffer space. This size then dynamically changes depending on how much space is available at any one time in the socket buffer. The space available is the original, or full window, size minus the amount of data queued in the socket buffer.

By advertising the full window in this manner, the present invention emulates the situation where the data stored in the socket receive buffer has been read by the application. This is only done when the `SP_MSGWAITALL` flag is set which guarantees that the `recv()` thread is blocked waiting for a sufficient amount of data. When the application is not looking at this socket at all, such as when the application is busy doing something else

Docket No. AUS920010389US1

or when the system is busy, the operations of the prior art are implemented, i.e. the window is reduced as the buffer fills up.

Once the TCP input processing determines that there
5 is at least an amount of data in the socket receive
buffer equal to the value of `so_rcvlen`, the TCP input
processing will wake up the `recv()` thread. The `recv()`
thread is modified to reset the `SP_MSGWAITALL` flag and
copy an amount of data equal to `so_rcvlen` in the socket
10 receive buffer to the application's buffer. The next time
the user calls `recv()` with `MSG_WAITALL`, if there is
partial data stored in the socket receive buffer, a
window update is sent out to the sending computing
device. This will let the sender resume sending in case
15 it is waiting for the window to be opened, i.e. Waiting
for the receiver to inform the sender that there is space
available in the receive socket buffer.

Figure 4 is a flowchart outlining an exemplary
operation of the present invention. The functions
20 described in **Figure 4** may be performed in software,
hardware, or a combination of software and hardware. In
a preferred embodiment, the functions of the present
invention detailed in **Figure 4** are performed as software
instructions executed on a processor, such as those shown
25 in **Figures 2** and **3**.

As shown in **Figure 4**, the operation starts with
invoking the `recv()` function (step **410**). The value for
`so_rcvlen` is set (step **420**). A determination is then
made as to whether there is an amount of data stored in
30 the socket receive buffer at least equal to the value of

Docket No. AUS920010389US1

so_rcvlen (step **430**). If so, an amount of data equal to so_rcvlen is copied to the application buffer (step **500**).

If there is not at least an amount of data equal to so_rcvlen in the socket receive buffer, the SP_MSGWAITALL flag is set (step **440**). The operation then waits for a data segment to be received in the socket receive buffer (step **450**). A determination is made as to whether a data segment is received in the socket receive buffer (step **460**). If not, the operation returns to step **450** and continues to wait for a data segment.

Upon receiving a data segment in the socket receive buffer, a determination is made as to whether there is at least an amount of data in the socket receive buffer equal to so_rcvlen (step **470**). If so, then the recv() thread is awoken (step **480**) and the SP_MSGWAITALL flag is reset (step **490**). An amount of data equal to so_rcvlen is then copied from the socket receive buffer to the application buffer, i.e. the application that called the recv() function (step **500**).

If there is not at least an amount of data equal to so_rcvlen in the socket receive buffer, the modified TCP input and output processing are performed (step **510**). This includes not waking up the recv() thread and sending out acknowledgments for every alternate segment received. In addition, the full window is advertised to the sender of the data segment. The operation then returns to step **450** and waits for the next data segment to arrive at the socket receive buffer.

Thus, the present invention provides a mechanism for performing bulk read operations using a socket receive buffer that avoids the problems associated with the prior

Docket No. AUS920010389US1

art. The present invention moves the monitoring functions from the application layer `recv()` function to the TCP layer and thus, eliminates the need to repeatedly awaken and put back to sleep the `recv()` function. This provides a considerable savings with regard to processing cycles due to the reduction in overhead. In addition, the present invention provides acknowledgment messages for every alternate segment and thus, the problems associated with the non-transmission of acknowledgment messages in the prior art are avoided. Moreover, the full window is advertised to senders of data segments so that the delay associated with the prior art is avoided.

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media such as a floppy disc, a hard disk drive, a RAM, and CD-ROMs and transmission-type media such as digital and analog communications links.

The description of the present invention has been presented for purposes of illustration and description, but is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in

Docket No. AUS920010389US1

the art. The embodiment was chosen and described in
order to best explain the principles of the invention,
the practical application, and to enable others of
ordinary skill in the art to understand the invention for
5 various embodiments with various modifications as are
suited to the particular use contemplated.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2